

27. The object data entries are referenced by a corresponding pointer 26, so for instance the pointer stored in pointer memory space "A" references the address of corresponding pointer data "A" stored in the pointer memory data space. In this manner any references made to the pointer data "A" are referenced to within the pointer memory space by pointer "A".

[0022] Pointers are sequentially written into the pointer memory space starting in proximity of the start address 28 and progressing towards the end address 29. Therefore in the pointer memory space 31, pointer "A" is at a lower address than pointer "B". Object data on the other hand is encoded in a converse manner. The first set of object data 33 corresponding to pointer "A" is encoded starting in proximity of the memory end address 29, where data is sequentially written from a higher address to a lower address – from an end of the data object to a beginning such that the data object remains in a same order compatible with PKCS15. Therefore, in the pointer data memory space, pointer data "A" is at a higher address than pointer data "B".

[0023] A pointer data start address 23 is not provided since the object data starts at the memory end address 29 and progresses towards the memory start address 28. Though data storage is described in this fashion, typically, objects are written in the same forward direction as currently employed by PKCS15 systems. The object start locations are determined based on object size and a last available data location such that the objects are placed at the end of the available data space.

[0024] Encoding data within the directory file 25 in this manner results in a single block of unused memory 32 having no data contained therein. This single unused block of memory 32 is located between the last encoded pointer located within the pointer memory space 31 and the last set of encoded object data located within the object data memory space 33.

[0025] As a result, there is no longer a limitation in placing the pointer data start address 23, since it is no longer required, as well as no longer wasting storage space due to insufficient space in only one of the two memory areas - pointer data and object data.

This single continuous block of unused memory 32 within the directory file enables encoding applications to make better use of this memory space contained within the directory file, eliminating the non-sequential empty memory space within the directory file.

[0026] Advantageously this format is backwards compatible with PKCS15 reading and encoding processes because the pointers within the pointer memory space still address the corresponding information stored within the pointer data memory space. With the improved method more information can be encoded within each directory file since the improved method of encoding facilitates improved memory management within the directory file.

[0027] For example the total amount of non-volatile memory allocated to the DODF is 100 bytes. The data objects stored in the DODF are either 19 bytes, or 39 bytes in length. If there are five data pointers to data objects stored within the DODF and five 19 byte data objects stored within the DODF then the total space of 100 bytes within the DODF is utilized effectively in the prior art system. If for instance there are three data pointers to data objects, a 19 byte data object and two 39 byte data object stored within the DODF then again then the total space of 100 bytes within the DODF is utilized effectively in the prior art system. In the prior art if the number of pointers to data objects within the DODF is fixed to five upon initialization of the directory file, then only five 19 byte data objects are written into the DODF, or one 39 byte data object and two 19 byte data objects, which leaves 18 bytes of unused data within the directory file with no data written to therein, since the data objects are minimum 19 bytes in size. However according to the present invention, the limitation in the amount of pointer space would not be present since the pointer data space and object data space share the same continuous piece of memory within the DODF. Advantageously in the invention if the size of the data object needs to be changed then the data object can be custom tailored to suit the exact amount of free memory within the DODF since the pointer data and data objects share the same memory space.

[0028] Numerous other embodiments may be envisaged without departing from the spirit or scope of the invention.